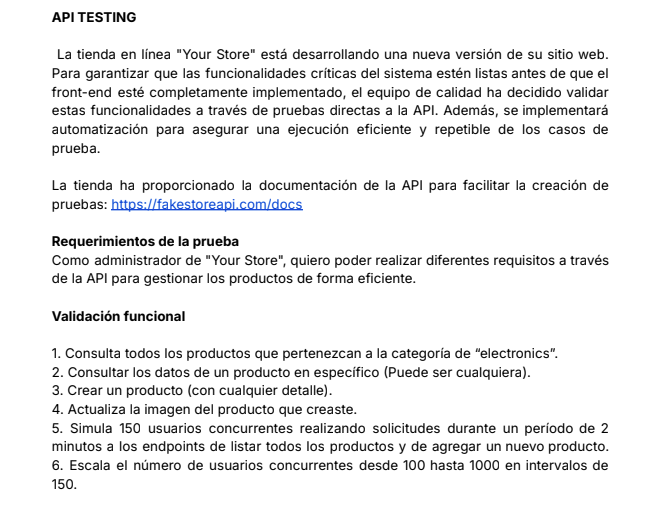
API TESTING



**Validación Funcional**

Se realizaron las siguientes validaciones:

1. **Consulta de productos por categoría “electronics”**

* Se validó la respuesta del API utilizando Python.
* Se verificó que el código de estado retornado fuera 200 (OK).
* Se comprobó que el resultado fuera una lista de elementos.
* Se validó que la lista contuviera al menos un elemento.
* Se confirmó que los productos pertenecieran a la categoría electronics.
* Se verificó que cada producto incluyera el campo title.

**2. Consulta de un producto específico**

* Se verificó que la respuesta retornara el código de estado 200 (OK).
* Se comprobó que el campo id correspondiera al producto consultado.
* Se validó el valor del precio.
* Se verificó la cantidad disponible del producto.

**3. Creación de un producto**

* Se validó que la petición retornara un código de estado 200 (OK).
* Se comprobó la correcta creación del producto con los datos enviados.
* Se verificó que los datos del producto creado existieran en la respuesta.

**4. Actualización de la imagen del producto creado**

* Se validó que la petición retornara un código de estado 200 (OK).
* Se comprobó que los datos del producto fueran actualizados correctamente.
* Se verificó posteriormente la eliminación del producto

**Elementos necesarios**

🔹 ***1. Instalar Python***

* Descarga Python 3.9 o superior desde [python.org](https://www.python.org/downloads/" \t "_new).
* Durante la instalación marca la opción ✅ “Add Python to PATH”.
* Verifica instalación desde la terminal: python –version

***2. Crear entorno virtual (recomendado)***

* Esto mantiene tus dependencias organizadas.

# Windows

python -m venv venv

venv\Scripts\activate

# Mac/Linux

python3 -m venv venv

source venv/bin/activate

***3. Instalar librerías necesarias***

* **Maven** (si usarás Maven para dependencias)

Verificar desde la terminal si esta instaldo:

mvn -version

Las básicas para API Testing son:

pip install requests pytest

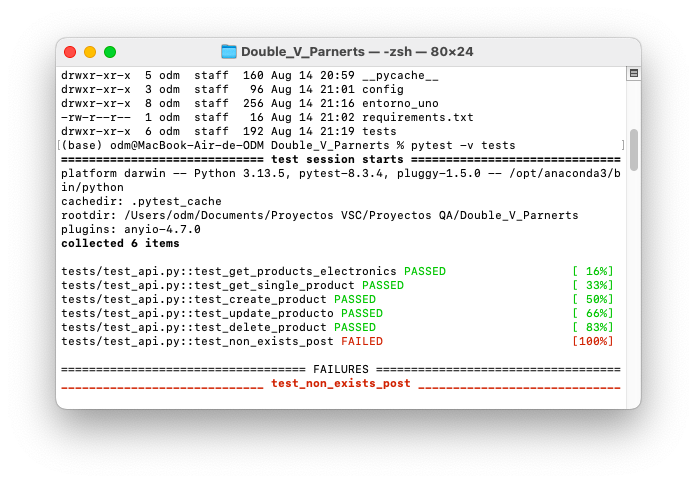
Explicación:

* **requests** → para hacer peticiones HTTP (GET, POST, PUT, DELETE).
* **pytest** → framework de pruebas automatizadas.

***Pasos para ejecutar el proyecto***

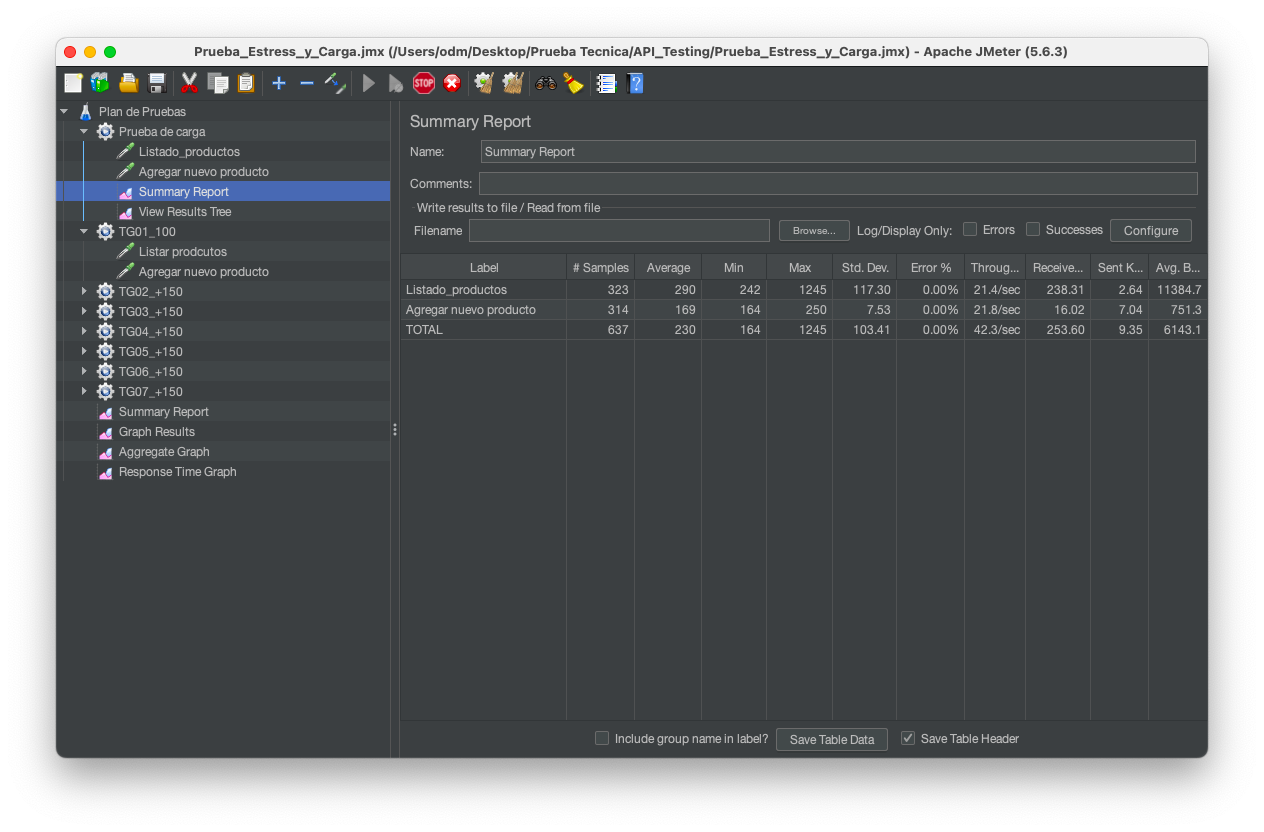
1. Copiar la carpeta Double\_V\_Parnerts dentro de su equipo
2. Abrir una terminal y ubicarse en el proyecto Double\_V\_Parnerts
3. Verificar que tenemos intsalado python y pytest
4. Escribimos en la terminal python --version ó python3
5. Escribimos pytest –version
6. Ejecutamos los casos de prueba pytest – v tests
7. Tests es la carpeta donde se encuentran todos los casos a ejecutar

Resultado de pruebas ejecutadas

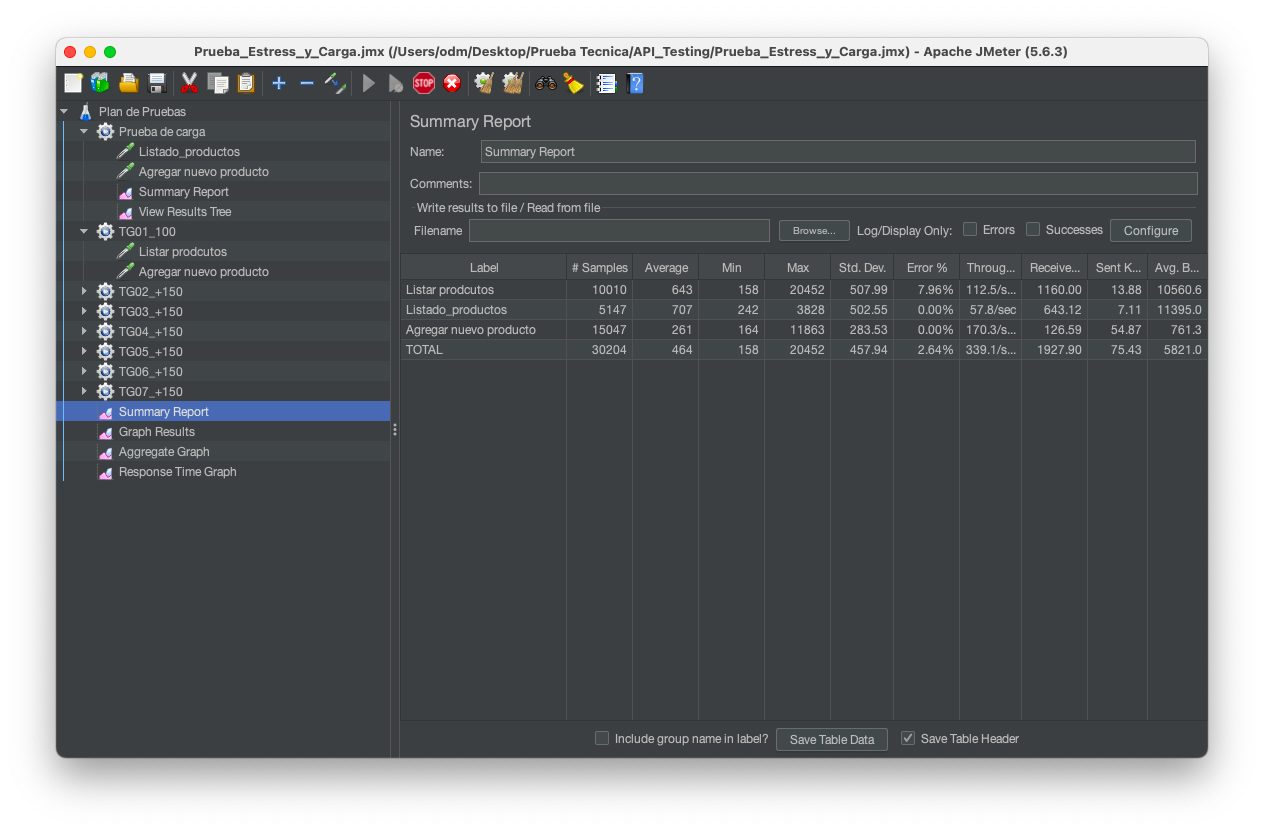


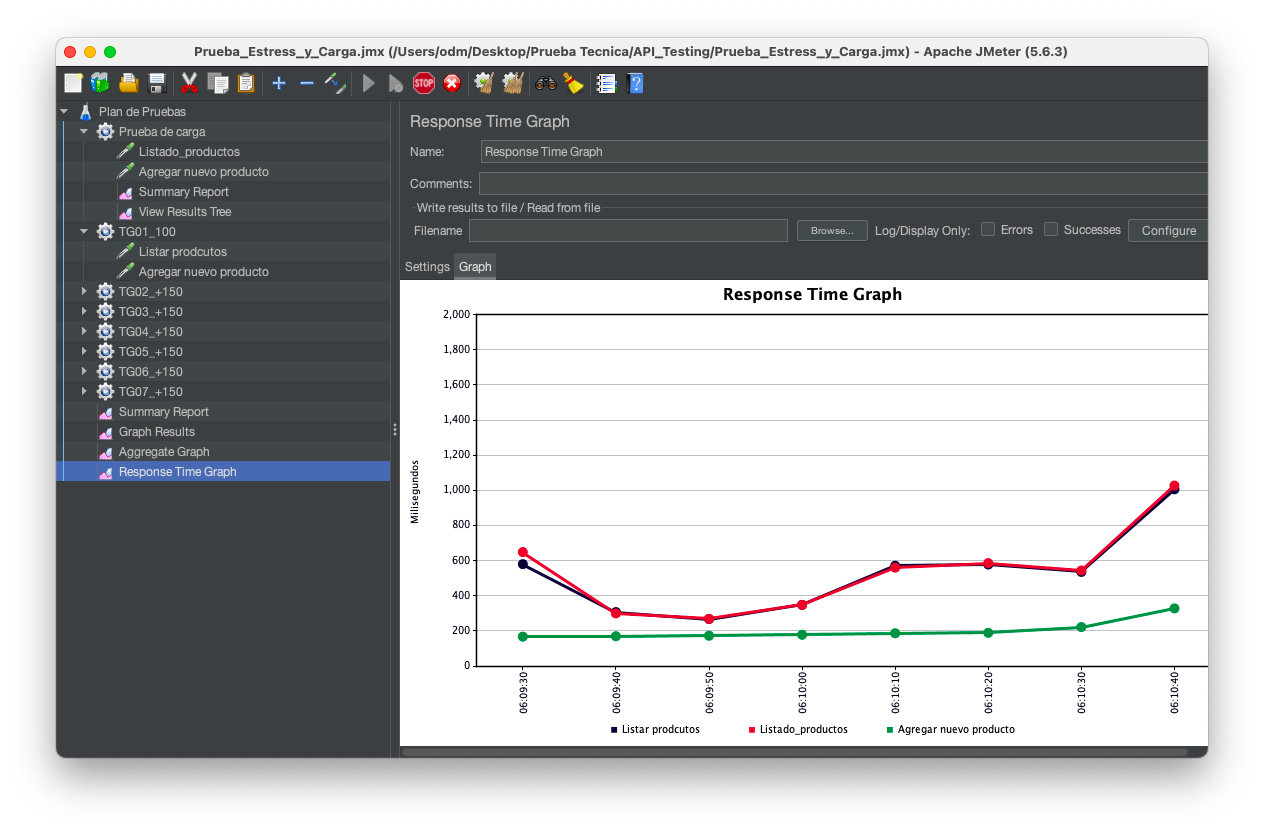
Se creo la estructura de las peticiones en Jmeter y se hicieron las pruebas de estress y carga

5. Simula 150 usuarios concurrentes realizando solicitudes durante un período de 2 minutos a los endpoints de listar todos los productos y de agregar un nuevo producto.



6. Escala el número de usuarios concurrentes desde 100 hasta 1000 en intervalos de 150.





Se adjunta archivo generado



**Funcionalidades que se Automatizaron**

1. Completa el formulario de registro.
2. Valida el inicio de sesión.
3. Prueba el restablecimiento de contraseña.
4. Ve a la sección de Laptops & Notebooks y selecciona la opción “Show all laptops &notebooks”
5. Agrega al carrito de compras un portátil MacBook Pro.
6. Utiliza la barra de búsqueda para encontrar una tablet Samsung Galaxy y agregala al
7. carrito de compras.
8. Elimina la MacBook Pro del carrito.
9. Agrega otra unidad de la tablet Samsung Galaxy.
10. Completa el proceso de compra hasta la confirmación de la orden.

* Se implementaron los scripts de automatización utilizando Selenium y TestNG
* Se uso un lenguaje de programación Java.
* Aplica patrones de diseño cómo Page Object Model.
* Genera un reporte de las pruebas ejecutadas – En el video se muestra el reporte generado
* Se adjunta video de las pruebas automatizadas

***Requisitos***

**1. Requisitos previos**

* Java JDK (8 o superior) → [Descargar](https://www.oracle.com/java/technologies/javase-downloads.html" \t "_new).

Verificar desde la terminal si esta instaldo:

java -version

* **Maven** (si usarás Maven para dependencias)

Verificar desde la terminal si esta instaldo:

mvn -version

* **IDE**: IntelliJ IDEA o Eclipse.

**2. Verificar versión de Google Chrome**

Antes de instalar ChromeDriver, necesitas saber qué versión de Chrome tienes:

* Mac / Windows:  
  Abre Chrome → Menú (⋮) → Ayuda > Información de Google Chrome.  
  Ejemplo: Versión 116.0.5845.140.

👉 ChromeDriver debe coincidir con tu versión de Chrome (o ser compatible con esa "major version").

**3. Instalar en Windows**

Opción A: Manual (más común)

1. Ve a: https://sites.google.com/chromium.org/driver/
2. Descarga el zip que coincida con tu Chrome.
3. Descomprime → tendrás chromedriver.exe.
4. Copia chromedriver.exe en una ruta conocida, por ejemplo:

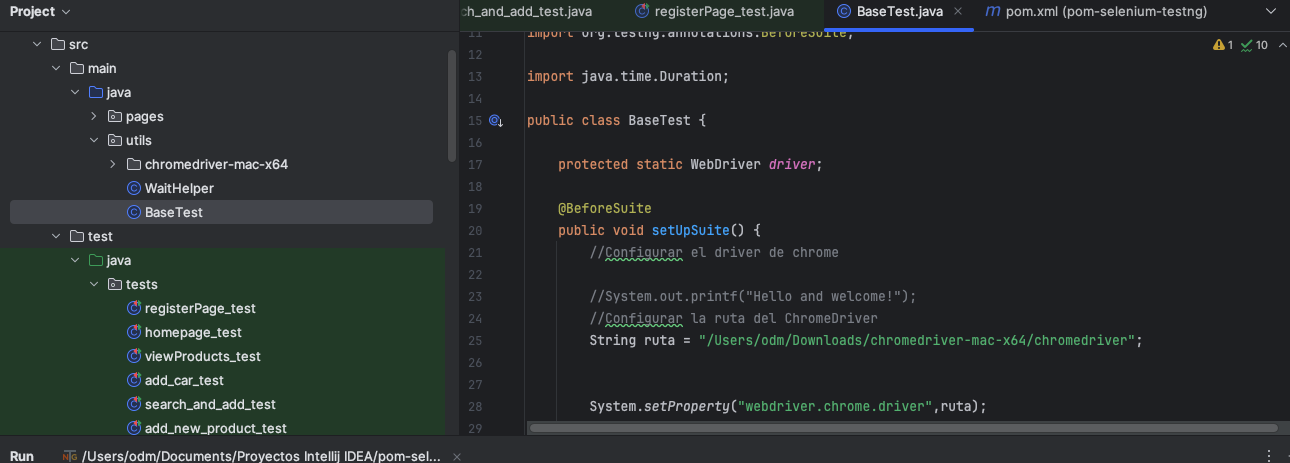
C:\chromedriver\

Agrega esa ruta al **PATH**:

* Inicio → “Editar las variables de entorno del sistema”.
* En **Variables del sistema** → selecciona Path → Editar.
* Agrega: C:\chromedriver\

Modificar la ruta de donde se ubica el chromedriver de su equipo en el archivo

src/java/utils/BaseTest

Modificar el campo ruta con la de tu chomedriver

Indicaciones generales

1. Crea un único repositorio en GitHub donde subas la solución de toda la prueba. Dentro

de este repositorio, organiza las dos partes del desafío (Punto 1 y Punto 2) en carpetas

separadas, asegurándose de que cada parte esté bien estructurada y documentada,

indicando lo necesario para clonar el proyecto y ejecutarlo.

2. Si para el punto 1 utilizas algún framework o herramienta específica, asegúrate de que

quede claro qué dependencias se necesitan instala

La prueba la tendrás que enviar al correo: welcome@doublevpartners.com

paula.correa@doublevpartners.com con el asunto “Prueba Técnica + tu nombre”.

¡¡¡Mucha suerte y gracias por el tiempo que nos dedicas!!! :)

¡Manos a la